Soluciones Ejercicios Tema 9

Germán Moltó Martínez
gmolto@dsic.upv.es
Estructuras de Datos y Algoritmos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Implementación de ColaPrioridad con ListaConPI (I)

```
public class ListaOrdenada<E extends Comparable<E>> implements
   ColaPrioridad<E> {
  protected ListaConPI<E> lpi;
   public ListaOrdenada(){
          this.lpi = new LEGListaConPI<E>();
   }
   public void insertar(E x){
    this.lpi.inicio();
    while ( !this.lpi.esFin() && this.lpi.recuperar().compareTo(x) <= 0 ){
          this.lpi.siguiente();
     }
    this.lpi.insertar(x); }
```


Implementación de ColaPrioridad con ListaConPI (II)

```
public E eliminarMin(){
   lpi.inicio();
   E elMinimo = lpi.recuperar();
   lpi.borrar();
   return elMinimo;
}
public E buscarMin(){
   this.lpi.inicio();
   return this.lpi.recuperar();
}
public boolean esVacia(){  return lpi.esVacia(); }
}
```


Uso de Cola de Prioridad en Impresión (1/2)

```
public class TrabajoImpresion implements Comparable<TrabajoImpresion> {
   private int nHojas;   private long timestamp;
   public TrabajoImpresion(int nHojas){
     this.nHojas = nHojas;  this.timestamp = System.currentTimeMillis();
   }
   public int compareTo(TrabajoImpresion t){
     int diff = nHojas - t.nHojas;  if (diff != 0) return diff;
     if (this.timestamp < t.timestamp) return -1;
     else if (this.timestamp > t.timestamp) return 1;
     else return 0;
    }
  public String toString(){ return "Trabajo de " + nHojas + " hojas ."+timestamp;}
 }
```

Uso de Cola de Prioridad en Impresión (1/2)

```
public class MiDriverImpresora implements DriverImpresora{
  ColaPrioridad<TrabajoImpresion> cp;
  public MiDriverImpresora(){
     cp = new LPIColaPrioridad<TrabajoImpresion>();
   }
   public void recibirTrabajoImpresion(TrabajoImpresion t){
      cp.insertar(t);
   }
   public TrabajoImpresion elegirTrabajoImpresion(){
      return cp.recuperarMin();
   }
}
```


Análisis de Secuencias de ADN

```
public class SecuenciaADN implements
   Comparable<SecuenciaADN>{
    private String secuencia;
   private String descripción;
    public SecuenciaADN(String secuencia){
        this.secuencia = secuencia;
   }

public int compareTo(SecuenciaADN o){
    return this.secuencia.length() – o.secuencia.length();
 }

 }
```


Ordenar Arrays con Cola de Prioridad (1/2)

```
public static <E extends Comparable<E>> void
   ordenarAscen(E[] v){
      int i;
      ColaPrioridad<E> c = new LPIColaPrioridad<E>();
      for (i = 0; i < v.length; i++) c.insertar(v[i]);
      i = 0;
      while (!c.esVacia()) {v[i] = c.eliminarMin(); i++;}
    }
```


Ordenar Arrays con Cola de Prioridad (2/2)

```
public static <E extends Comparable<E>> void
   ordenarDescen(E[] v){
      ColaPrioridad<E> c  = new LPIColaPrioridad<E>();
      ordenarDescen(v, 0, c);
   }
public static <E> void ordenarDescen(E[] v, int i,
   ColaPrioridad<E> c){
      if ( i < v.length){
            c.insertar(v[i]);
            ordenarDescen(v, i+1, c);
            v[i] = c.eliminarMin();
      } }
```

Número de Repeticiones en una Frase

```
public static int numPalabrasRepetidas(String texto){
  Diccionario<String, String> d = new ABBDiccionario<String,String>();
  String [] tokens = texto.split(" ");
   int cont = 0;
   for (int i = 0; i < tokens.length; i++){
       try{
            d.recuperar(tokens[i]);
            cont++;
       }catch(ElementoNoEncontrado ex){
           d.insertar(tokens[i],"");
        }
    }
    return cont; }
```


Palabras Repetidas y Frecuencia en una Frase

```
public static String palabrasRepetidasFrec(String texto){
   Diccionario<String, Integer> d = new ABBDiccionario<String,Integer>();
   String token; int frecuencia;
   String [] tokens = texto.split(" ");
   for (int i = 0; i<tokens.length; i++){
     token = tokens[i];
      try{
         frecuencia = d.recuperar(token).intValue();
         frecuencia++;
      }catch(ElementoNoEncontrado ex1){  frecuencia = 1; }
      d.insertar(token, new Integer(frecuencia));
   }
   return d.toString(); }
```


Traducción en Diccionario Bilingüe

```
public static String traducir(String frase, Diccionario<String,String> d){
        String fraseTraducida = "";
        String[] palabrasFrase = frase.split ( " " );
        for ( int i = 0; i < palabrasFrase.length; i++ ){
           try{
              Sring trad = d.recuperar(palabrasFrase[i].toLowerCase());
              fraseTraducida += trad + " ";
            } catch(ElementoNoEncontrado e) {
                  fraseTraducida += " --- ";
               } }
        return fraseTraducida; }
```